

Use Case Study

02/July/2004

Bill Tanenbaum
US-CMS/Fermilab



Outline

- **Very Brief Review of Reconstructed Objects**
- **Statement of Problem**
- **Solution of Problem**



July 02, 2004

Bill Tanenbaum U.S.CMS/Fermilab

2

Reconstructed Objects

- Stored persistently in DST
- Each object type must inherit from **RecObj**
- Objects are hierarchical



July 02, 2004

Bill Tanenbaum U.S.CMS/Fermilab

3

Collections of Reconstructed Objects

□ Defined by:

- Type of object (e.g. Track, Vertex, Jet)
- Name of algorithm (must be unique)
- Version of algorithm
- Parameter Set
- Component Set
 - Dependencies on other (lower level) objects
- Calibration Set

□ All these define the **RecConfig**

- The same **RecConfig** gives reproducible results

□ **RecQuery**

- User friendly way to define **RecConfig**



July 02, 2004

Bill Tanenbaum U.S.CMS/Fermilab

4

Statement of Problem

Actor: Physicist

Goal: Write a reconstruction algorithm to calculate the total energy deposited in the calorimeters for an event. Make the energy persistent and write it into the DST.

Trigger: Each event



Comments on the Solution

□ Class CaloEnergy

- Existing class **CaloRecHit** for reconstructed calorimeter hits
- **CaloRecHit** does not (currently) inherit from **RecObj**
- Therefore, **CaloRecHit** reconstruction is non-standard
- Write class “**CaloEnergy**” that sums energies from **CaloRecHits**

□ Class CaloEnergyBuilder

- Algorithm for constructing **RecCollection<CaloEnergy>**
- Only one object in collection. Total calorimeter energy for event.
- Has no components (empty ComponentSet) because **CaloRecHit** reconstruction is non-standard.
- I used two parameters for illustrative purposes
- CalibrationSet not yet implemented in framework



Persistency considerations

□ Class CaloEnergy

- Must be persistence capable
- CaloEnergyBuilder and RecCollection<CaloEnergy> are transient!
- CaloEnergy needs POOL(SEAL) data dictionary
- In a src directory, two files are needed: classes_def.xml and classes.h

□ classes_def.xml contents

```
<lcgdict>
  <class name="CaloEnergy"/>
</lcgdict>
```

□ classes.h contents

```
#include "Utilities/Configuration/interface/Architecture.h"
#include "Calorimetry/CaloEnergy/interface/CaloEnergy.h"
```



Code for solution

□ Color coded for this presentation

- Blue is for boilerplate code (same for all algorithms) – very little effort
- Black is for the algorithm specific names –little effort
- Red is algorithm specific (the hard part)
- Green is just for printout – not needed



The CaloEnergy Class

```
#ifndef CaloEnergy_h
#define CaloEnergy_h
#include "CARF/Reco/interface/RecObj.h"
class CaloEnergy : public RecObj {
public:
    CaloEnergy() : energy_(0) {}
    explicit CaloEnergy(double e) : energy_(e) {}
    virtual ~CaloEnergy(){}
    double energy() const {return energy_;}
private:
    double energy_;
};
#endif
```



July 02, 2004

Bill Tanenbaum U.S.CMS/Fermilab

The Algorithm Header

```
#ifndef CaloEnergyBuilder_h
#define CaloEnergyBuilder_h
#include "CARF/Reco/interface/RecAlgorithm.h"
class CaloEnergy;
class CaloEnergyBuilder : public RecAlgorithm<CaloEnergy> {
public:
    static RecConfig defaultConfig();
    explicit CaloEnergyBuilder(const RecConfig& config = defaultConfig());
    virtual ~CaloEnergyBuilder();
    void reconstruct();
private: // Algorithm parameters
    double HCALcutoffEnergy;
    double ECALcutoffEnergy;
};
#endif
```



July 02, 2004

Bill Tanenbaum U.S.CMS/Fermilab 10

The Algorithm Source

```
#include "Utilities/Configuration/interface/Architecture.h"
#include "Calorimetry/CaloRecHit/interface/CaloRecHit.h"
#include "Calorimetry/CaloEnergy/src/CaloEnergyBuilder.h"
#include "Calorimetry/CaloEnergy/include/CaloEnergy.h"
#include "Calorimetry/CaloCommon/interface/CaloItr.h"
#include "CARF/Reco/interface/ParameterSetBuilder.h"
#include "CARF/Reco/interface/ComponentSetBuilder.h"
#include "CARF/Reco/interface/RecConfig.h"
// Constructor goes here
CaloEnergyBuilder::~CaloEnergyBuilder() {}
// defaultConfig() goes here
// reconstruct() goes here. THIS IS THE GUTS
#include "Utilities/Notification/interface/PackageInitializer.h"
#include "CARF/Reco/interface/RecBuilder.h"
static PKBuilder<RecBuilder<CaloEnergyBuilder>> calo_energy_builder_factory
    ("CaloEnergyBuilderBuilder");
```



The Algorithm Constructor

```
CaloEnergyBuilder::CaloEnergyBuilder(const RecConfig& config) :  
    RecAlgorithm<CaloEnergy>(config) {  
    setMeAsDefault();  
  
    // initialize the reconstruction parameters  
    HCALcutoffEnergy= parameter<double>("HCALcutoffEnergy");  
    ECALcutoffEnergy=parameter<double>("ECALcutoffEnergy");  
  
    std::cout << std::endl << "CaloEnergyBuilder:HCALcutoffEnergy = " << HCALcutoffEnergy <<  
        std::endl << "CaloEnergyBuilder:ECALcutoffEnergy = " << ECALcutoffEnergy << std::endl ;  
}
```



The Algo. defaultConfig() method

RecConfig

```
CaloEnergyBuilder::defaultConfig() {  
  
    Name name("CaloEnergyBuilder");  
    Version version("1.0");  
  
    ParameterSetBuilder param_set_builder;  
    const double default_tolerance = 0.001; // tolerance for parameter comparison  
    param_set_builder.addParameter("HCALcutoffEnergy",0.0,default_tolerance);  
    param_set_builder.addParameter("ECALcutoffEnergy",0.0,default_tolerance);  
  
    ComponentSetBuilder component_set_builder;  
  
    return RecConfig(name,version,param_set_builder.result(),component_set_builder.result());  
}
```



July 02, 2004

Bill Tanenbaum U.S.CMS/Fermilab

13

The Algo. reconstruct() method

```
void CaloEnergyBuilder::reconstruct() {
    double energy=0.0;
    CaloItr<CaloRecHit> hcalhit(Suld("HR","HCAL","01"));
    while(hcalhit.next()) {
        if (hcalhit->energy() > HCALcutoffEnergy) energy += hcalhit->energy();
    }
    CaloItr<CaloRecHit> ebryhit(Suld("CR","EBRY","01"));
    while(ebryhit.next()) {
        if (ebryhit->energy() > ECALcutoffEnergy) energy += ebryhit->energy();
    }
    CaloItr<CaloRecHit> efryhit(Suld("CR","EFRY","01"));
    while(efryhit.next()) {
        if (efryhit->energy() > ECALcutoffEnergy) energy += efryhit->energy();
    }
    addObjToReconstructor(new CaloEnergy(energy));
} // addObjToReconstructor adds the object to the collection and takes ownership
```



The addition to writeDST::analysis()

```
// Called once per event
// Other reconstructed objects not shown
void writeDST::analysis() {
    if (theCaloEnergyCollection == 0) {
        theCaloEnergyCollection = new RecCollection<CaloEnergy>( RecQuery
            ("CaloEnergyBuilder", "1.0"));
    }
    cout << "Found " << theCaloEnergyCollection->size() << " CaloEnergy." << endl;
    double energy = 0.0;
    for(RecCollection<CaloEnergy>::const_iterator it=theCaloEnergyCollection->begin();
        it!=theCaloEnergyCollection->end(); ++it) {
        energy += (*it)->energy();
    }
    cout << "Total energy" << setw(8) << setprecision(4) << energy << " GeV" << endl;
}
```

